Universidad de los Andes
Ingeniería de Sistemas y Computación
ISIS 2203 Infraestructura Computacional
2018-10

**Caso de Estudio 3 – Análisis de Desempeño**
**Sistema de Gestión Empresarial y Operativa de una Compañía Transportadora**

**Objetivos**

- Evaluar las características de desempeño de la infraestructura computacional que soporta el despliegue de una aplicación.
- Simular condiciones de carga y configuraciones de software con el fin de analizar el comportamiento de una aplicación.
- Evaluar el efecto que tienen los mecanismos de seguridad sobre el desempeño de una aplicación.

**Descripción General**

En esta tercera parte del caso se aborda la problemática del desempeño de la infraestructura computacional para el despliegue de la aplicación trabajada en el caso dos: el sistema de rastreo de unidades de distribución. En el caso dos se trataron los requerimientos de canales seguros. En esta parte del caso se explorarán las relaciones entre el desempeño de la aplicación y las características del software, ante diferentes niveles de carga. Además se estudiarán los efectos que tiene el uso de mecanismos para ofrecer canales seguros en el desempeño de las máquinas.

**Problemática**

En el despliegue de una solución computacional es de vital importancia tener en cuenta la infraestructura subyacente con el fin de ofrecer un desempeño adecuado durante el ciclo de vida de la solución.

**Actividades**

1. **Preparación.** Vamos a simular escenarios para analizar el comportamiento del servidor y de los recursos de la máquina antes diferentes niveles de carga. Para ello deberá instalar el servidor y el cliente de la aplicación en máquinas diferentes; podría usarse la misma máquina pero esto introduce distorsiones en las mediciones. El servidor le será suministrado y el cliente será el que usted mismo desarrolló para el Caso 2.

    1.1. Revise el código del servidor que maneja concurrencia por medio de un pool con un número fijo de threads (más adelante se presentan los diferentes tamaños para el pool). A diferencia del primer proyecto, para implementar el pool no usamos las primitivas de java, usamos la librería de Java java.util.concurrent.Executors, clases ExecutorService y Executors.

    1.2. Los indicadores de eficacia son aquellos atributos que describen qué tan bien se desempeña la aplicación para el usuario final. Por ejemplo, tiempo de respuesta, número de transacciones atendidas por unidad de tiempo y número de transacciones perdidas. Además, están los indicadores de uso de los recursos, como porcentaje de uso de procesador, de memoria y de disco, así como número de conexiones de red activas.

    Diseñe e implemente mecanismos para medir los siguientes indicadores: (1) tiempo para obtener la llave simétrica (desde la lectura del socket, incluyendo procesamiento, hasta la creación de la llave simétrica), (2) tiempo de respuesta a una actualización (tiempo entre la salida del mensaje del cliente - ACT1- y la llegada de la respuesta del servidor -OK-) y (3) número de transacciones perdidas dada una carga determinada sobre el sistema.

    1.3. Explore la herramienta de medida de desempeño de Windows para determinar porcentaje de uso de CPU, memoria y red. Para Monitorear los recursos de la infraestructura siga el manual “Manual Performance Monitor V2.0”.

    1.4. En Sicua+ encontrará los tutoriales de Gload, la herramienta para generar carga. Instale y explore la herramienta.

2. **Identificación de la plataforma**. Defina la máquina en la que correrá el servidor e identifique las siguientes características. Tenga en cuenta que el servidor deberá correr en la misma máquina para todos los experimentos (si cambia de máquina los resultados no serán comparables).

    - Arquitectura (32 o 64 bits)

- Número de núcleos (*cores*)
- Velocidad del procesador
- Tamaño de la memoria RAM
- Espacio de memoria asignado a la JVM

3. **Comportamiento de la aplicación con diferentes estructuras de administración de la concurrencia**. Este escenario tiene como objetivo evaluar cambios en el comportamiento del servidor ante diferencias en el manejo de la concurrencia, para esto:

   - Revise el código del servidor para que implemente un pool de threads de tamaño definido y asigne threads por conexión.
   - Genere escenarios diferentes cambiando el valor de las variables número de threads en el pool y carga. Para la primera variable (número de threads) use los valores 1, 2 y 8. Para la segunda variable (carga) use 400 transacciones iniciadas con retardos de 20 ms, 200 transacciones iniciadas con retardos de 40 ms y 80 transacciones iniciadas con retardos de 100 ms. (*)
   - Para cada uno de los 9 escenarios posibles (3 tamaños de pool * 3 tipos de carga) mida los siguientes indicadores: tiempo para creación de la llave simétrica, tiempo para actualización, número de transacciones perdidas y porcentaje de uso de la CPU en el servidor. Repita cada experimento 10 veces.
   - Cree las siguientes gráficas:
     a. Fije Carga y genere: # threads vs. tiempo de creación de la llave (3 cargas diferentes = 3 gráficas, 3 tamaños de pool -1, 2 y 8-: 3 conjuntos de puntos por gráfica)
     b. Fije Carga y genere: # threads vs. tiempo de actualización (mismo número de gráficas)
     c. Fije Carga y genere: # threads vs. # de transacciones perdidas (mismo número de gráficas)
     d. Fije Carga y genere # threads vs. porcentaje de uso de la CPU en el servidor (mismo número de gráficas)
   - Para cada gráfica analice, compare con otras gráficas y escriba sus conclusiones sobre el comportamiento del sistema ante las diferentes configuraciones.

   *(*) ATENCIÓN*: Tenga en cuenta que al aumentar el tamaño del pool, aumenta el consumo de memoria por parte de la aplicación dentro de la JVM. Por este motivo la aplicación podría superar el tamaño de memoria asignado por defecto dentro de la máquina virtual y no correría adecuadamente. Para resolver esta situación modifique el valor asignado por defecto:

   - Primero haga clic derecho en la clase principal del servidor / Run As / Run Configurations.
   - En la pestaña de argumentos diríjase a "Program arguments" y escriba: -Xms<memory_size> donde <memory_size> es el tamaño de la memoria en bytes que le desea asignar al programa.

4. **Comportamiento de la aplicación ante diferentes niveles de seguridad**.
   4.1. Repita los experimentos del punto 3 (para los indicadores 2, 3 y porcentaje de uso de CPU), usando un servidor y un cliente sin seguridad.
   4.2. Responda: ¿Cuál es el resultado esperado sobre el comportamiento de una aplicación que implemente funciones de seguridad vs. una aplicación que no implementa funciones de seguridad?
   4.3. Genere las gráficas b, c y d del punto 3, e indique si ellas confirman los resultados esperados (en el punto anterior). Justifique su respuesta.

**Entrega.** Archivo zip o tar con:

1. Código fuente del servidor y del cliente (debe incluir las modificaciones para medida de indicadores).
2. Archivo (o archivos) con los datos recopilados (un archivo por escenario, o una hoja Excel por escenario).
3. Informe con: (a) Descripción detallada de los cambios realizados para medir los indicadores, incluyendo el código (cliente o servidor) y líneas dónde se hizo el cambio, (b) identificación de la plataforma, (c) respuestas al punto 3 (todas las gráficas y los comentarios), y (d) respuestas al punto 4 (todas las gráficas y los comentarios).

**Referencias**


- *Java Threads (Third Edition)*. Oaks, S., & Wong, H. O'Reilly, 2004.
- *Presentaciones del curso (Análisis de desempeño).*